Doxygen
Tutorial

Resumen

Doxygen es una herramienta para generar documentación a partir del código fuente. Es un sistema de documentación para C++, C, Java, ObjectiveC, Python, IDL (Corba and Microsoft flavors), Fortran, VHDL, PHP, C#.

Índice

1. Instalación

Primero necesita obtener la última versión de Doxygen de la página web:
http://www.doxygen.org
Además usted necesitará:

- Las herramientas GNU flex, bison, GNU make y strip.
- Para generar un Makefile necesitará perl.
- El script de configuración asumirá que usted posee las herramientas de Unix como sed, date, find, uname, mv, cp, cat, echo, tr, cd y rm.

Para hacer uso de todas las características de Doxygen, deberá instalar tambien las siguientes herramientas:

- Una distribución de LaTeX, por ejemplo teTeX 1.0, esto es necesario para generar las salidas Latex, PostScript y PDF.
- Graphviz, Graph visualization toolkit, versión 1.8.10 o mayor. Necesaria para incluír los gráficos.
- Un intrprete de ghostscript.
- Python para genera la documentación propia de doxygen.

2. Compilación

Compilación:
1. *gunzip doxygen$VERSION.src.tar.gz* descomprime el archivo.
2. *tar xf doxygen$VERSION.src.tar* desempaqueta el archivo.
3. ejecute el script configure: *sh ./configure*
Para conocer todas las opciones para usar el configure ejecute:
configure help
Compile el programa ejecutando:
make

El programa debería compilar sin problemas y tres binarios deberían crearse en el directorio bin de la distribución.

Opcional: Generar el manual del usuario ejecutando:
make docs

Luego de compilar las fuentes ejecute un make install para instalar doxygen.

3. Comenzar

El ejecutable doxygen es el programa principal que analiza el código y genera la documentación.

El ejecutable doxytag es necesario solo si desea generar referencias a documentación externa de la que no tiene las fuentes.

De manera opcional, puede ser usado el ejecutable doxywizard, que es una interfaz grafica para editar el archivo de configuración.

El siguiente gráfico muestra la relación entre las herramientas y el flujo de información entre ellas:

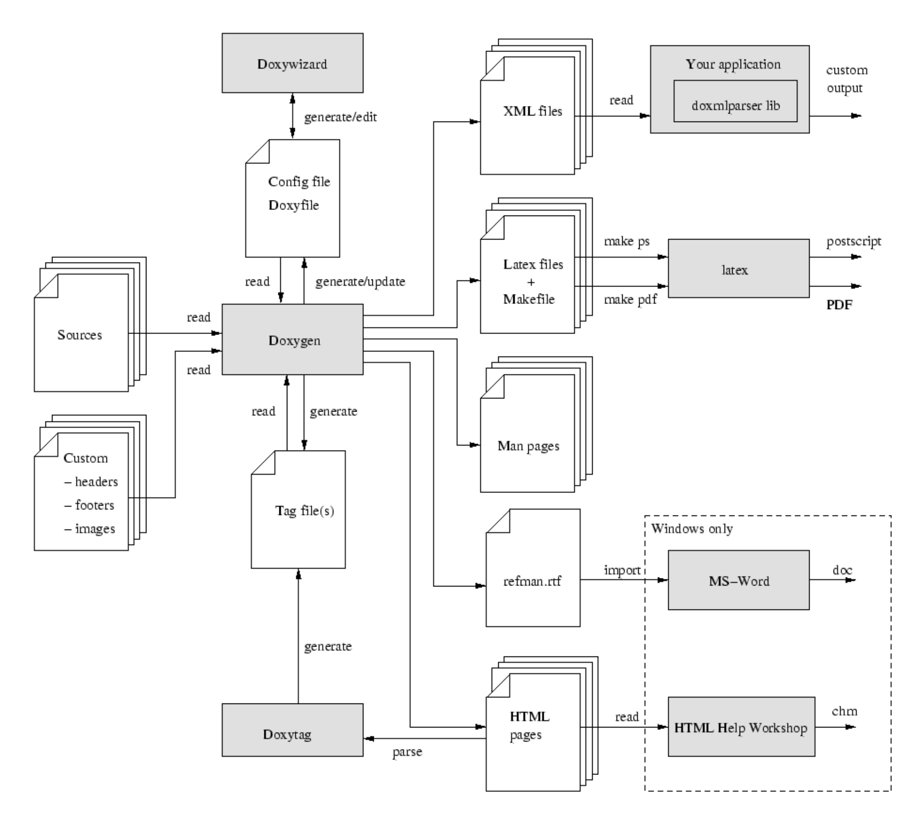


Figura 1: *flujo de informacion*

1

4. Crear un archivo de configuración

Doxygen usa un archivo de configuración para determinar todas sus opciones. Cada proyecto debería tener su propio archivo de configuración.

Para simplificar la creación de este archivo, doxygen puede crear uno con el comando:

doxygen g configfile

donde <configfile>es el nombre del archivo de configuración, si se omite el nombre del archivo, se creará uno llamado doxyfile. Si ya existe un archivo con el mismo nombre, doxygen renombrará el actual a <configfile>.bak antes de generar el nuevo.

El archivo de configuración tiene un formato similar al de un Makefile; consiste en un numero de asignaciones (tags) de la forma:

TAGNAME = VALUE

Si comienza a usar doxygen para un proyecto existente, puede hacerse una idea de como es la estructura y de como resultará la documentación poniendo la tag EXTRACT_ALL = YES*.*

Si no desea editar el archivo de configuración con un editor de texto, puede usar doxywizard, una interfaz gráfica que puede crear, leer y editar los archivos de configuración del doxygen de manera gráfica y sencilla.

Para pequeños proyectos de pocos C/C++ fuentes y headers, puede dejar la tag INPUT vacía y doxygen buscará por fuentes en el directorio actual.

Si tiene un proyecto mas grande debe asignar los directorios o directorio raíz a la tag INPUT*, y agregar uno o mas patrones de archivos (*.c, *.h, etc.)a la tag*FILE_PATTERNS *. Sólo los archivos que coinciden con estos patrones serán analizados. Para un análisis recursivo de un árbol de fuentes, la tag*RECURSIVE *debe tener valor YES.*

5. Generando la documentación

Para generar la documentación debe ingresar:

doxygen configfile

Dependiendo de sus opciones, doxygen creará los directorios html, rtf, latex, xml y/o man.

El directorio de salida por defecto es en el que doxygen se ha iniciado. El directorio raiz en el cual se escribe la salida de doxygen puede modificarse usando

OUTPUT_DIRECTORY . El directorio especifico de formato dentro del directorio de salida puede ser seleccionado usando las *HTML_OUTPUT*, *RTF_OUTPUT*,

LATEX_OUTPUT, *XML_OUTPUT*, y *MAN_OUTPUT* tags del archivo de configuración.

5.1. Salida HTML

La documentación HTML generada puede verse apuntando un browser html al archivo index.html en el directorio html. Para mejores resultados es mejor utilizar un browser con soporte para CSS. Algunas características como*GENERATE_TREEVIEW* requieren de un browser con soporte para DHTML y Javascript. Si tiene planeado usar el buscador (tag *SEARCHENGINE*), debe ver la salida html via un web server que soporte PHP (ej:apache con el modulo PHP instalado).

5.2. Salida LaTeX

La documentación latex generada debe ser compilada anteriormente por un compilador latex. Para simplificar el proceso de compilación, doxygen crea un Makefile dentro del directorio latex. Los contenidos del Makefile van a depender de las opciones de *USE_PDFLATEX*, si está deshabilitado, tipeando make en el directorio latex generará un archivo dvi llamado refman.dvi. Este archivo puede verse usando xdvi o convertirlo a PostScript refman.ps tipeando make ps (esta acción requiere la herramienta dvips).

También es posible convertir los archivos a PDF si tiene instalado el intérprete de ghostscript, con el comando make pdf (o *make pdf_2on1*). Para obtener mejores resultados en la salida PDF deberia setear los tags*PDF_HYPERLINKS y USE_PDFLATEX* en yes. En este caso el Makefile contendra sólo el destino para construir refman.pdf directamente.

5.3. Salida RTF

Doxygen combina la salida RTF a un único archivo llamado refman.rtf. Este archivo es optimizado para importar desde Microsoft Word. Cierta

información es codificada usando campos. Para mostrar el valor real usted necesita seleccionar todo (Edit Select All) y luego alternar campos.

5.4. Salida XML

La salida XML consiste de dump estructurado de información reunida por doxygen. Cada componente tiene su propio archivo XML y tambn existe un archivo index llamado index.xml.

Un archivo XSLT llamado combine.xslt tambin es generado y puede ser usado para combinar todos los archivos XML a un único archivo.

Doxygen tambin genera dos archivos esquema XML index.xsd (para el archivo index) und.xsd (para los archivos de componentes). Este archivo de esquemas describe los posibles elementos, sus atributos y como están estructurados.

5.5. Salida Man

las paginas de manual generadas pueden verse utilizando el programa man. Debe asegurarse que el directorio man este en el man path (ver la variable de entorno MANPATH).

6. Documentando el código

Si la opción *EXTRACT_ALL* tiene el valor NO en el archivo de configuración (opción por defecto), doxygen sólo generará documentación para miembros documentados, archivos, clases, etc. Para documentar hay dos opciones:

- Colocar un bloque especial de documentación delante de la declaración o definición de los miembros, clases, etc.

- Colocar un bloque especial en algún otro lugar (otro archivo u otra ubicación) y poner un "comando estructural" en el bloque de documentación. Un comando estructural enlaza un bloque de documentación a una cierta entidad que puede ser documentada.

Los archivos solo pueden ser documentados de la segunda forma. Las funciones, variables, typedefs, defines, no necesitan un comandodas las líneas vacías resultantes son tratadas como separadores de párrafos.

Esto evita que se deban agregar comandos de nuevo párrafo para lograr una documentación mas legible.

· Los links son creados para palabras que corresponden a clases documentadas (a menos que la palabra es precedida por %, en este caso la palabra no será enlazada y el signo % será eliminado).

· Los links a miembros son creados cuando se encuentran ciertos patrones en el texto.

· Los marcadores HTML que están en la documentación son interpretados y transformados en equivalente.

7. Bloques especiales de documentación

Es un bloque de documentación de C o C++ con algunas marcas adicionales, así doxygen puede reconocerlo y generar su documentación.

Para cada ítem del código existen dos (a veces tres) tipos de descripciones que juntas forman la documentación: una descripción brief (breve) de una línea y una descripción detailed (detallada) mas extensa. Para todos y funciones tambin existe una descripción llamada in body (en cuerpo), que consiste en la concatenación de todos los bloques de documentación encontrados en el cuerpo de la función.

Existen varias formas de marcar un bloque de comentario como una descripción detallada:

JavaDocs alternativa es usar un bloque de al menos dos lineas de comentarios de C++, donde cada línea comienza con un / o ! adicional

```
///
/// comentarios.
///
```

o bien

```
//!
//! comentarios
//!
```

Para aquellos a quienes desean comentarios mas visibles en la documentación pueden utilizar:

```
/******************************//**
*.comentarios..
*********************************/

o bien


//////////////////////////////////
/////comentarios..
//////////////////////////////////
```

Para las descripciones breves (brief) tambin existen varias posibilidades:

Puede usar el comando*brief* con alguno de los bloques de comentarios anteriores.

Este comando termina al final del párrafo, así la descripción detallada continúa luego de una línea en blanco.

/*!\brief descripción breve.

* continua la descripción brermina en el primer punto seguida de un espacio o una nueva línea.

```
/** Descripción breve que termina aquí. Descripción
* detallada continua aquí.
*/

o bien


/// Descripción breve que termina aquí. Descripción detallada
/// continua aquí.
```

Doxygen sólo permite una descripción breve y una detallada.

7.1. Documentando miembros

Si desea documentar los miembros de un archivo, estructura, union, class o enum debe colocar un marcador <en el bloque:

```
int var; /*!< descripción después del miembro */

int var; /**< descripción detallada después del miembro */

int var; //!<descripción detallada después del miembro
       //!<

int var; ///< descripción detallada después del miembro
       ///<
```

Generalmente se deseará colocar una describpción breve luego de un miembro, esto se realiza de la siguiente manera:

```
int var; //!< descripción breve después del miembro
```

```
O bien

int var; ///< Descripción breve después del miembro
```

Nótese que estos bloques tienen la documentación de los miembros delante de la definición (incluyendo funciones globales). De sta manera la documentación puede ser ubicada en el archivo fuente en lugar del archivo header. Esto permite que el archivo de headers quede compacto, y permite a los implementadotes de los miembros un acceso mas directo a la documentación.

Tambien las descripciones breves pueden situarse antes de la declaración y la descripción detallada antes de la definición del miembro.

Éstos bloques solo pueden ser utilizados para documentar miembros o parámetros, no pueden ser usados para documentar archivos, clases, uniones, estructuras, grupos, namespaces y enums. Por lo tanto, los comandos estructurales mencionados en la próxima sección (como *class*) son ignorados dentro de stos bloques de comentarios.

8. Documentación en otros sitios

Hasta ahora hemos asumido que los bloques de documentación están siempre ubicados al frente de una declaración o definición de un archivo, clase o namespace o delante o detrás de un miembro. Puede haber ocasiones en las que se desee comentar en otros sitios.

Doxygen le permite colocar bloques de documentación prácticamente en cualquier sitio, la excepción es dentro del cuerpo de una función o dentro de un bloque de comentarios normal en C style.

Lo que se necesita es un comando estructural dentro del bloque de documentación, lo que conlleva a algunas duplicaciones de información. Por lo tanto en la práctica es recomendable evitarstos comandos a menos que otros requerimientos lo fuercen.

Los comandos estructurales (como todos los otros comandos) comienzan con un backslash (\), o un arroba ()

Para documentar una función global en C, typedef, enum o definición del preprocesador, primero debe documentar el archivo que lo contiene (usualmenteste será un archivo header).

9. Listas

Doxygen provee varias maneras de crear listas de ítems:

9.1. Usando signos menos

Colocando una cantidad de signos menos () alineados en una columna al principio de la línea generarán automáticamente una lista. Las listas nuricas tambien pueden ser generadasusando un signo menos seguido por un símbolo numeral (#). Es posible anidar listas indentando los ítems.

Si utiliza tabs para la indentación dentro de listas, por favor asegúrese que *TAB_SIZE* en el archivo de configuración tenga puesto el valor correcto.

Puede finalizar una lista comenzando un nuevo párrafo o colocando un punto (.) en una línea vacía en el mismo nivel de indentación que la lista que desea terminar.

Veamos un ejemplo:

```
/**
 * Texto anterior a la lista
 * - item lista 1
 *   - sub item 1
 *     - sub sub item 1
 *     - sub sub item 2
```

```
*     .
*     El punto de arriba significa que termina la lista
*     de sub sub items.
*     Mas texto del primer sub item.
*   .
*   El punto de arriba termina la lista de sub items.
*   Mas texto para el primer item de la lista.
*   - sub item 2
*   - sub item 3
* - list item 2
* .
* Mas texto en el mismo parrafo.
*
* Mas texto en un nuevo parrafo.
*/
```

9.2. Usando comandos HTML

Si lo desea puede usar comandos HTML dentro de los bloques de documentación. Utilizar estos comandos tiene la ventaja de ser un mtodo más natural para los ítems de listas que consisten de múltiples párrafos.

10. Agrupar

Doxygen posee tres mecanismos para agrupar. Un mecanismo trabaja a nivel global, creando una nueva página para cada grupo. Estos grupos son llamados modulos en la

documentación. El segundo mecanismo trabaja dentro de una lista de miembros de alguna entidad compuesta y es llamado grupos de miembros (member groups). Para las páginas existe un tercer mecanismo llamado subpaginar (subpaging).

10.1. Módulos

Módulos es una manera de agrupar cosas en páginas separadas. Se puede documentar un grupo como un todo, así tambin como todos los miembros individuales. Los miembros de un grupo pueden ser archivos, namespaces, clases, funciones, variables, enums, typedefs y defines, pero tambin otros grupos.

Para definir un grupo debe colocarse el comando \defgroup en un bloquero de un grupo específico colocando un comando \ingroup dentro de su bloque de documentación.

Para evitar colocar comandos \ingroup en la documentación por cada miembro, se puede agrupar miembros abriendo un marcador antes del grupo y el marcador de cierre despus del grupo.

Los grupos tambin pueden ser anidados usando stos marcadores de grupos.

10.2. Grupos de miembros

Si un compuesto (clase o archivo) tiene muchos miembros, es usualmente deseado agruparlos.

Un grupo de miembros es definido de la siguiente manera:

```
//@\{

\...

//@}
```

bloque ó

```
/*@{*/

\...
```

```
/*@}*/
```

Antes de abrir un marcador de bloque, puede colocarse un bloque de comentarios separado. Este bloque debe contener el comando name (o \name) y se utiliza para especificar el encabezado del grupo.

No está permitido el anidamiento de grupos de miembros.

10.3. Subpaginar

La información puede ser agrupada en páginas usando los comandos \page y \mainpage.

Normalmente, esto resulta en una lista plana de páginas, donde la página main es la primera en la lista.

10.4. Incluír fórmulas

Doxygen permite agregar fórmulas a la salida (funciona solamente para las salidas y HTML).

Son necesarias las siguientes herramientas:

- LaTeX: el compilador de LaTeX.
- Dvips: Una herramienta para convertir archivos DVI a archivos PostScript.
- Gs: el intrprete GhostScript para convertir archivos PostScript a bitmaps.

  Existen tres formas de incluir fórmulas en la documentación:

1. Utilizando fórmulas que aparecen en el texto en una línea. Éstas fórmulas deben colocarse entre un par de comandos \f$ como por ejemplo:

   ```
   La distancia entre \f$(x_1,y_1)\f$ y \f$(x_2,y_2)\f$

   es \f$\sqrt{(x_2x_1)^2+(y_2y_1)^2}\f$.


   La salida se verá así:
   ```

   La distancia entre (x_1, y_1) y (x_2, y_2) es $\sqrt{(x_2x_1)^2 + (y_2y_1)^2}$.

2. Fórmulas no numéricas en líneas separadas deben colocarse entre los comandos \f[y \f]

   por ejemplo:

```
\f[
   |I_2|=\left| \int_{0}^T \psi(t)
         \left\{
            u(a,t)
            \int_{\gamma(t)}^a
            \frac{d\theta}{k(\theta,t)}
            \int_{a}^\theta c(\xi)u_t(\xi,t)\,d\xi
         \right\} dt
      \right|
\f]
```

La salida se verá así:

$|I_2| = \left| \int_0^T \psi(t) \left\{ u(a,t) \int_{\gamma(t)}^a \frac{d\theta}{k(\theta,t)} \int_a^\theta c(\xi) u_t(\xi,t)\, d\xi \right\} dt \right|$

3. Las fórmulas u otros elementos de que no son del ambiente matemático pueden ser especificados usando el comando \fenvironment, donde environment es el nombre del ambiente de , el fin del comando es \f} .

   Veamos un ejemplo, equation array:

```
\f{eqnarray*}{

  g &=& \frac{Gm_2}{r^2} \\
    &=& \frac{(6.673 \times 10^{11}\,\mbox{m}^3\,
         \mbox{kg}^{1}\,
        \mbox{s}^{2})(5.9736 \times 10^{24}\,
         \mbox{kg})}{(6371.01\,\mbox{km})^2} \\
    &=& 9.82066032\,\mbox{m/s}^2
\f}
```

La salida se verá así:

$g = \frac{Gm_2}{r^2}$
$= \frac{(6{,}673 \times 10^{-11}\,\mathrm{m}^3\,\mathrm{kg}^{-1}\,\mathrm{s}^{-2})(5{,}9736 \times 10^{24}\,\mathrm{kg})}{(6371{,}01\,\mathrm{km})^2}$
$= 9{,}82066032\,\mathrm{m/s}^2$

11. Gráficos y diagramas

Doxygen puede crear gráficos, generados por la herramienta dot de Graphviz (http://www.graphviz.org). La generación de gráficos en Doxygen generalmente se realiza automaticamente basada en las opciones del archivo de configuración. Estas opciones con excepción de DOT_PATH pueden tener los valores YES o NO. Estas son opciones globales para el proyecto. Los gráficos sólo serán generados si la herramienta dot está instalada, y la opción HAVE_DOT está en YES en el archivo de configuración.

- HAVE_DOT indica que la herramienta dot está disponible.
- DOT_PATH es el "path.ª la herramienta dot, si no está en $PATH.
- GRAPHICAL_HIERARCHY generará una representación grafica de la jerarquía del código. Actualmente está solo disponible en HTML.
- INCLUDE_GRAPH generará un gráfico por cada archivo documentado que incluya al menos un archivo más. Esta característica está disponible sólo para HTML y RTF.
- CALL_GRAPH generará un gráfico por cada función, sus llamadas, y las llamadas a ésta. Asímismo si *CALLER_GRAPH* está en YES se crearán los gráficos mostrando por quien son llamadas las funciones.
- CALLER_GRAPH generará gráficos que indica las funciones llamadas por la función documentada.
- COLLABORATION_GRAPH generará un gráfico por cada clase documentada y estructura que muestre las relaciones de uso con otras estructuras.

  También se puede utilizar la sintaxis de dot para generar un gráfico. Los comandos específicos de dot deben estar entre los comandos \dot y \enddot.

12. Generación automática de links

La mayoría de los sistemas de documentación tienen una sección especial "see also"(vea también), donde pueden insertarse links a otras partes de la documentación. Además de que Doxygen posee un comando para comenzar una sección así (vea \sa), permite colocar esta clase de links en caulquier parte de la documentación. Para la documentación en LaTeX, en lugar de un link se escribe el número de página. Además el índice al final del documento puede utilizarce para encontrar rapidamente la documentación de una clase, miembro, namespace o

archivo. Para las páginas man no se generan referencias. Las próximas secciones muestran como generar links a las entidades documentadas en un archivo fuente.

12.1. Links a páginas web y direcciones de mail

Doxygen reemplazará automáticamente las URLs y direcciones de mail que encuentre en la documentación por links (en HTML).

12.2. Links a archivos

Todas las palabras que contengan un punto (.) que no sea el último caracter de la palabra son considerados nombres de archivos. Si la palabra es el nombre de un archivo documentado de entrada, se generará automáticamente un link a la documentación de ese archivo.

12.3. Links a funciones

Se crearán links a funciones si es encontrado uno de los siguientes patrones:

```
1. <functionName>"("<argument-list>")"
2. <functionName>"()"
3. "::"<functionName>
4. (<className>"::")n<functionName>"("<argument-list>")"
5. (<className>"::")n<functionName>"("<argument-list>")"<modifiers>
6. (<className>"::")n<functionName>"()"
7. (<className>"::")n<functionName>
con n>0
```

12.4. Links a variables, typedefs, enum types, enum values y defines

Todas éstas entidades pueden ser referidas de la misma manera que se explica en la sección anterior. Para mayor claridad es recomendable utilizar los patrones 3 y 7 en éste caso.

12.5. typedefs

Los typedefs que involucran clases, estructuras y uniones, como typedef struct StructName TypeName, crean un alias para StructName, entonces se crearán links para StructName, cuando StructName mismo o TypeName son encontrados.

13. Comandos

Doxygen provee un gran número de comandos especiales, comandos XML y comandos HTML, que pueden ser usados para realzar o estructurar la documentación dentro de un bloque de comentario. Si por alguna razón existe la necesidad de definir un nevo comando, se puede utilizar una definicion ALIAS. La definición de un alias debe ser especificada en el archivo de configuración usando el tag de configuración ALIASES.

13.1. Alias simples

La forma mas simple de un alias es la sustitución simple de la forma:

name=value

Por ejemplo:

```
ALIASES += sideeffect="\par Side Effects:\n"
```

esto permitirá poner el comando \sideefect en la documentación, que resultará en un párrafo definido por el usuario con encabezado Side Effects: Nótese que puede colocar \n en la parte del valor de un alias para insertar una nueva línea.

13.2. Alias con argumentos

Los alias pueden tener uno o más argumentos. En la definición del alias se necesitan un número específico de argumentos entre llaves. En la parte del valor de la definición se pueden colocar marcadores \x, donde x representa el número de argumento empezando por 1. Por ejemplo:

```
ALIASES += l{1}="\ref \1"
```

Dentro de un bloque de comentarios:

```
/** See \l{SomeClass} for more information. */
```

ó

```
/** See \ref SomeClass for more information. */
```

```
ALIASES += l{1}="\ref \1"
```

```
ALIASES += l{2}="\ref \1 \"\2\""
```

```
/** See \l{SomeClass,Some Text} for more information. */
```

Dentro de un bloque de comentarios se expanderá a:

/** See \ref SomeClass ”Some Text”for more information. */

donde el comando con un sólo argumento seguirá funcionando como se muestra anteriormente.

Alias también pueden expresarse en términos de otros alias. Ejemplo un nuevo comando \reminder puede ser expresado como \xrefitem por un comando intermediario \xreflist:

```
ALIASES += xreflist{3}="\xrefitem \1 \"\2\" \"\3\" " \
ALIASES += reminder="\xreflist{reminders,Reminder,Reminders}" \
```

Para alias con mas de un argumento se utliza la coma como separador. Si se desea poner una coma dentro del comando necesitará usar un backslash, ejemplo:

\lSomeClass,Some text\, with an escaped comma

13.3. Anidando comandos

Puede utilizar comandos como argumentos de alias, incluyendo comandos definidos utilizando alias, ej:

```
ALIASES += Bold{1}="<b>$\backslash$1</b>"
ALIASES += Emph{1}="<em>$\backslash$1</em>"
```

dentro de un bloque de comentarios:

```
/** This is a $\backslash$Bold{bold $\backslash$Emph{and} Emphasized}
 text fragment. */
```

y se expandirá de la siguiente manera:

```
/** This is a <b>bold <em>and</em> Emphasized</b> text fragment. */
```

14. Links a documentación externa

Si su proyecto depende de librerías externas o herramientas, hay varias razones por las cuales no incluír todas la fuentes en cada pasada de doxygen.

<++>Espacio en disco: Alguna documentación puede estar disponible fuera del directorio de salida de doxygen, por ejemplo en algún lugar de la web. Si lo desea, puede vincular a estas páginas en lugar de generar la documentación en su directorio local de salida. Velocidad de Compilación: Los proyectos externos suelen tener una frecuencia de actualización diferente a las de su propio proyecto. No tiene mucho sentido dejar que doxygen analice las fuentes de éstos proyectos externos una y otra vez, incluso si nada ha cambiado. Memoria: Para árboles muy grandes de código fuente, permitir que doxygen analice todas las fuentes puede tomar demasiado de la memoria del sistema. Dividiendo las fuentes en varios ”packages”, las fuentes de un paquete puede ser analizadas por doxygen, mientras que todos los otros ”packages”que dependen de este paquete, estan linkeados externamente. Esto ahorra una gran cantidad de memoria. Disponibilidad: Para algunos proyectos que están documentados con doxygen, las fuentes pueden simplemente no estar disponibles. Cuestiones de derecho de autor: Si el paquete externo y su documentación tiene derechos de autor de otra persona, tal vez sea mejor - o incluso necesario - referenciarlo en lugar de incluir una copia del mismo con la documentación de su proyecto. Cuando el autor prohíbe la redistribución, esto es necesario. Si el autor exige el cumplimiento de alguna condición de licencia como condición de redistribución, y usted no quiere que se obligan a cumplir con esas condiciones, en referencia a su copia de su documentación es preferible a que se incluya una copia.

Si se aplican algunas de las razones anteriores, puede utilizar el mecanismo de archivo de marcadores (tag file) de doxygen. Un tag file es una representación compacta de las entidades encontradas en las fuentes externas. Doxygen puede leer y generar tag files.

Para generar un tag file debe poner el nombre de un tag file a continuación de la opción *GENERATE_TAGFILE* en el archivo de configuración.

Para combinar la salida de uno o mas proyectos externos con el propio, debe especificar el nombre de los tag files a continuación de la opción TAGFILES en el archivo de configuración.

Un tag file no contiene información sobre dónde se encuentra la documentación externa. Ésto podría ser un directorio o una URL. Entonces cuandos e incluye un tag file, debe especificar el lugar en que se encuentran. Existen dos formas de hacer esto:

- En tiempo de configuración: Sólo asigne el lugar de salida al tag file especificado en la opción TAGFILES. Si usa un path realativo, debe ser relativo respecto del directorio donde la salida HTML de su proyecto es generada.
- Luego de la compilación: Si no asigna un lugar a el tag file, doxygen generará dummy links para todas las referencias HTML externas. También generará un script de perl llamado installdox en el directorio de salida HTML. Éste script debe ejecutarse para reemplazar los dummy links por links reales para todos los archivos HTML generados.

En algunos (excepcionales) casos puede tener la documentación generada por doxygen, pero no los fuentes ni los tag files. En este caso puede usar la herramienta doxytag para extraer un tag file de los HTML generados. Otro caso en el que debe usar doxytag es si desea crear un tag file para la documentación Qt.

La herramienta doxytag depende de la estructura particular de la salida generada y de algunos marcadores especiales que son generados por doxygen. Ya que ésta clase de extracción es frágil y susceptible a errores es recomendable que sea usada en caso de que no exista otra alternativa.

15. Comandos Especiales

Todos los comandos en la documentación comienzan con un backslash (\)o un arroba (@). Algunos comandos tienen uno o mas argumentos. Cada argumento tiene un cierto rango:

- si <>entonces el argumento es una sola palabra.
- si () entonces el argumento se extiende hasta el final de la línea.
- si entonces el argumento se extiende hasta el próximo párrafo. Los párrafos están delimitados por una línea en blanco o por un indicador de sección.
- si [] entonces significa que el argumento es opcional.

Acontinuación se detalla una lista de los comandos más usados:

15.1. Formato de texto

- **\a <palabra>** escribe palabra en una fuente especial.Equivalente a \e.
- **\arg <palabra>** item de una lista, equivalente a .
- **\b <palabra>** escribe palabra en negrita.
- **\c <palabra>** escribe palabra en la fuente typewriter.
- **\code** Comienza una sección de bloque de código.
- **\endcode** Finaliza una sección de bloque de código (\code).
- **\copydoc ref** Copia documentación de ref.
- **\dot** Comienza un bloque de gráfico dot.
- **\enddot** Finaliza un bloque de gráfico dot.
- **\f$** Comienza y finaliza una fórmula de una línea.
- **\f[** Comienza un bloque de fórmula centrada.
- **\f]** Finaliza un bloque de fórmula centrada.
- **\image <formato><archivo>caption** Coloca una imagen dentro de la documentacion con la caption çaption".
- **\htmlonly** Comienza un bloque para salida HTML únicamente.
- **\endhtmlonly** Finaliza un bloque para salda HTML únicamente.
- **\n** Fuerza una nueva línea.
- **\verbatim** Comienza un bloque incluído como textual.
- **\endverbatim** Finaliza un bloque textual(\verbatim).

15.2. Indicadores Estructurales

- \ **addtogroup <nombre>[(titulo)]** Define un grupo igual que \defgroup, pero en contraste, usando <nombre>mas de una vez no resultara en un warning, sino en un grupo con la documentacion combinada y el primer titulo que encuentre en alguno de los comandos. El titulo es opcional, por lo tanto este comando tambien puede ser utilizado para agregar un numero de entidades a un grupo existente usando @ y @.
- **callgraph** Cuando este comando es colocado en un bloque de comentarios de una funcion y HAVE_DOT esta en Yes, doxygen generara un grafico de llamadas para esa funcion. El grafico sera generado a pesar del valor que tenga CALL_GRAPH.
- **callergraph** Cuando este comando es colocado en un bloque de comentarios de una funcion, y HAVE_DOT esta en Yes, doxygen generara un grafico de llamador de funciones para esa funcion. El grafico sera generado a pesar del valor que tenga CALLER_GRAPH.
- **class <nombre>[<archivo-header>] [<nombre-header>]** Indica que un bloque de comentarios contiene documentacion para una clase con el nombre <nombre>.
- **def <nombre>** Indica que un bloque de comentarios contiene documentacion para un macro #define.
- **defgroup <l><t>** Define un grupo de módulo con etiqueta 1 y título t.
- **enum <nombre>** Indica que un bloque de comentarios contiene documentacion para una enumeracion con nombre <nombre>.
- **example <nombre-archivo>** Indica que un bloque de comentarios contiene documentacion de un ejemplo de codigo fuente. El nombre del archivo fuente es <nombre-archivo>. EL texto de este archivo sera incluido en la documentacion, inmediatamente despues de la documentacion contenida en el bloque de comentarios. Todos los ejemplos son colocados en una lista. Puede incluirse parte del path en <nombre-archivo>en el caso de que el nombre no sea unico. Si es necesario mas de un archivo fuente para el ejemplo, puede utilizarse el comando **include**.
- **file [<nombre>]** Indica que un bloque de comentarios contiene documentacion para un fuente o archivo header con el nombre <nombre>. El archivo puede incluir (parte de) el "path" si no es unico. Si el nombre del archivo es omitido, entonces el bloque de documentacion que contiene el comando \file pertenecera al archivo en el que se encuentra. Importante: La doucmentacion de

funciones globales, variables, typedefs y enum solo seran incluidas en la salida si el archivo en el que se encuentran tambien esta documentado.

- **fn (declaracion de funcion)** Indica que un bloque de comentarios contiene documentacion de una funcion. Este comando solo es necesario si un bloque de comentarios no se encuentra delante o detras de la declaracion o definicion de la funcion. Si el bloque de comentarios esta delante de la declaracion, este comando puede (y para evitar redundancia deberia) ser omitido. Una declaracion de funcion completa, incluyendo argumentos debe ser especificada a continuacion del comando en una sola linea. Importante: No use este comando a menos que sea absolutamente necesario, ya que lleva a la duplicacion de informacion, y en consecuencia a errores.
- **hideinitializer** Oculta el valor por defecto de #define.
- **ingroup <l1>**[<**l2**>] Agrega documentacion al grupo 11.Multiples grupos pueden ser usados.
- **interface <n>** Bloque de documentaciön para la interface n.
- **internal** Todo el texto a continuación es sustraído hasta el final del bloque de comentarios.
- **mainpage (t)** El bloque de documentación del main o index.
- **name (header)** Bloque de documentación de grupo de miembros.
- **nosubgrouping** Desactiva el subagrupamiento (subgrouping).
- **overload (s**) Utilizado para documentar una funcion overloaded con firma s.
- **package <n>** Un bloque de documentación para el package n.
- **page <n>(t)** Indica un bloque de documentación de un grupo de página con etiqueta n y título t.
- **showinitializer** Muestra el valor por defecto de define.
- **struct <n>** Bloque de documentación para una estructura n.
- **typedef (t)** Bloque de documentación para un typedef t.
- **union <n>** Bloque de documentación para la union n.
- **var (v)** Bloque de documentación para la variable v.
- **weakgroup <n>** Equivalente a \addtogroup pero tiene menor prioridad en la resolución de conflictos de grupos.

15.3. Indicadores de sección

- **attention ...** Comienza un párrafo de Atención.
- **author loa** Incluye la lista de autores loa.
- **brief ...** Una breve descripción del elemento.
- **bug ...** Descripción de un bug.
- **cond <sec>** Comienza una sección condicional.
- **date ...** Incluye la fecha.
- **deprecated ...** Comienza un párrafo desaprobado.
- **else** Cláusula adicional al comando \if.
- **elseif <sec>** Cláusula adicional al comando \cond.
- **endcond** Finaliza una sección condicional.
- **endif** Finaliza una cláusula \if.
- **exception <e>** Comienza una descripción de la excepción para e.
- **if <sec>** Comienza una sección condicional.
- **ifnot <sec>** Comienza una sección condicional.
- **invariant ...** Comienza una decripción de un invariant.
- **note ...** Comienza un párrafo de notas.
- **par (t) ...** Comienza un párrafo con un título opcional t.
- **param <p>...** Comienza una descripción del parámetro p.
- **post ...** Comienza una descripción de una post-condición.
- **pre ...** Comienza una descripción de una pre-condición.
- **remarks ...** Comienza un párrafos de observaciones.
- **return ...** Comienza una descripción de valores de retorno.
- **retval <f>...** Describe el valor de retorno de una función f.
- **sa ref** Comienza un párrafo See Also.
- **see ref** Equivalente a \sa.
- **since ...** Describe desde cuando una entidad estuvo disponible.
- **test ...** Comienza un párrafo de descripción ”test case”.
- **throw <e>...** Equivalente a excepción.
- **todo ...** Comienza un párrafo To Do.
- **version ...** Comienza un párrafo donde puede ingresar la versión.
- **warning ...** Comienza un párrafo para mensajes de alerta.
- **xrefitem <k>(h) .** Crea un párrafo cross-referenced.

15.4. Indicadores de links

- **addindex (t)** Agrega t al index de LaTeX.
- **anchor (w)** Coloca un anchor invisible w en el documento.
- **endlink** Finaliza un link.
- **link <n>...** Crea un link a n con texto especificado por el usuario.
- **ref <n>(t)** Crea una referencia a n con texto t.
- **subpage <n>(t)** Crea una sub-página de nombre n.
- **section <l>(t)** Crea una sección en una página con etiqueta 1 y título t.
- **subsection <l>(t)** Crea una sub-sección en una página con etiqueta 1 y título t.
- **paragraph <n>(t)** Crea un párrafo en una página con etiqueta 1 y título t.

15.5. Bloques especiales de documentación en VHDL

Para VHDL un comentario comienza normalmente con "--". Doxygen extraerá los comentarios que comiencen con "--!". Existen solo dos tipos de bloques de comentarios en VHDL; de una línea "--!" que representa una descripción breve, y de más de una linea, hay que repetir "--!" por cada línea representa una descripción detallada.

Los comentarios siempre se ubican delante del ítem a documentar, con una excepción: para puertos el comentario puede también ir después del ítem, y luego es tratado como una descripción breve del puerto.

Aquí sigue un ejemplo de un archivo en VHDL con comentarios Doxygen:

```
-------------------------------------------------------
--! @file
--! @brief 2:1 Mux using with-select
-------------------------------------------------------

--! Use standard library
library ieee;
--! Use logic elements
    use ieee.std_logic_1164.all;

--! Mux entity brief description

--! Detailed description of this
--! mux design element.
entity mux_using_with is
    port (
        din_0   : in  std_logic; --! Mux first input
        din_1   : in  std_logic; --! Mux Second input
        sel     : in  std_logic; --! Select input
        mux_out : out std_logic  --! Mux output
    );
end entity;

--! @brief Architure definition of the MUX
--! @details More details about this mux element.
architecture behavior of mux_using_with is
begin
    with (sel) select
    mux_out <= din_0 when '0',
               din_1 when others;
```

```
end architecture;
```

Para obtener una salida apropiada debe setear la opción *OPTIMIZE_OUTPUT_VHDL* en YES en el archivo de configuración de Doxygen.

15.6. Otros

- \ % Este comando escribe el simbolo % a la salida.
- \>+ Este comando escribe el simbolo > a la salida.
- \<+ Este comando escrible el símbolo < a la salida.
- \# Este comando escribe el símbolo # a la salida.
- \$ Este comando escrible el símbolo $ a la salida.
- \& Este comando escribe el simbolo & a la salida.

16. Referencias

La información anterior se extrajo y tradujo del manual de Doxygen.
http://www.doxygen.org

Cecilia Chialchia -

ceciliach@iar.unlp.edu.ar